Kostengünstig? Aber sicher!

Bewertung von Architekturvarianten im Kontext von ISO 26262 und harter Echtzeit

Die Anzahl aktiver Sicherheits- und Assistenzsysteme mit hohen Verfügbarkeitsanforderungen und harten Echtzeitanforderungen nimmt stark zu. Werden die Echtzeitanforderungen erst spät betrachtet, resultieren daraus oft teure Änderungen an der System-, Software- und Hardware-Architektur. Wie diese vermieden werden können, zeigt ein konkretes Beispiel: Schon in der funktionalen Architektur werden sicherheitsrelevante Wirkketten identifiziert, End-to-End Echtzeitanforderungen zugewiesen und Zeitbudgets festgelegt. Aus der funktionalen Architektur können verschiedene Varianten technischer Architekturen abgeleitet und hinsichtlich ihrer Echtzeiteigenschaften simuliert und systematisch bewertet werden.

Als Fallbeispiel dient ein fiktives Fußgänger Airbag System (FABSY), das Fußgänger vor körperlichen Schäden bei der Kollision mit einem Auto schützen soll. Das System erkennt, wenn eine Kollision des Fahrzeugs mit einem Fußgänger unausweichlich ist, und löst in diesem Fall einen in der Fahrzeugfront angebrachten Fußgänger-Airbag aus.

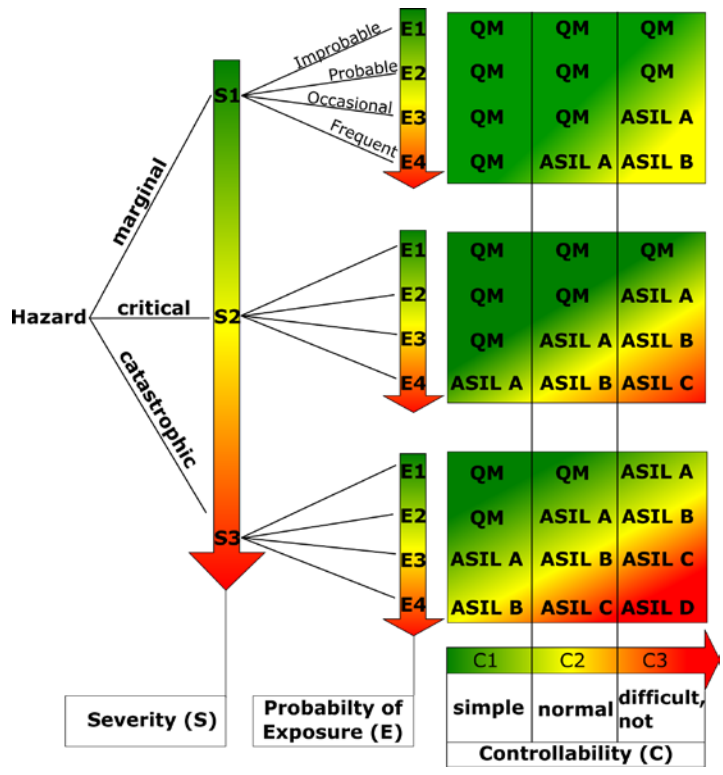

Bild 1: Risikograph zur Bestimmung des Automotive Safety Integrity Levels (ASIL). (© MethodPark)

Das System zeichnet sich durch Anforderungen an die funktionale Sicherheit und durch Echtzeitanforderungen aus: Für die Auslösung des Airbags steht nur ein enges Zeitfenster zur Verfügung, da der Airbag die gewünschte Schutzwirkung verfehlt, wenn er zu früh oder zu spät ausgelöst wird. Im Rahmen dieses engen Zeitfensters muss eine komplexe Wirkkette durchlaufen werden - von der Akquisition eines Bildes über die Bewertung der Situation bis hin zur Auslösung des Airbags. Kasten 1 „Echtzeitanforderungen" skizziert, wie Echtzeitanforderungen für das System abgeleitet werden.

In der Betrachtung der funktionalen Sicherheit muss neben der erwünschten Auslösung des Airbags auch berücksichtigt werden, dass durch eine unerwünschte Auslösung die Fahrzeuginsassen und weitere Verkehrsteilnehmer gefährdet werden können.

Gefahren- und Risikoanalyse

Das Ziel der Gefahren- und Risikoanalyse ist es, zunächst gefährliche Situationen, die vom zu entwickelnden System ausgehen könnten, zu identifizieren und zu kategorisieren. Diese Analyse wird im Teil 3 des geltenden Standards für funktionale Sicherheit im Automobil, der ISO 26262, beschrieben und bildet den Grundstein für die Erstellung des funktionalen Sicherheitskonzepts. Der ISO 26262 entsprechend werden aus den Ergebnissen dieser sogenannten G&R Analyse die jeweiligen Sicherheitsziele abgeleitet. Diese Sicherheitsziele, so der Standard, sind als Top-Level-Sicherheitsanforderungen zu interpretieren. Das prinzipielle Vorgehen der G&R folgt dem Grundsatz die Folgen eines

ID	Scenario considered	Mal function	Effect of failure	S		Comment for Severity	E		Comment for Exposure Time	C		Comment for Controllability	ASIL
B2	Fast driving within the city, pedestrian walks onto the road	Airbag does not inflate, false negative	Pedestrian collides with car	S3	Life-threatening injuries (survival uncertain), fatal injuries	life threatening or fatal injuries	E2	Low probability	pedestrians walking onto highway / freeway is possible but infrequent	C3	Difficult to controll or uncontrollable by skilled driver	situation cannot be controlled	B
B3	Slow driving within the city, pedestrian walks onto the road	Airbag does not inflate, false negative	Pedestrian collides with car	S2	Severe and life threatening injuries (survival probable)	severe and life threatening injuries	E2	Low probability	pedestrians walking onto highway / freeway is possible but infrequent	C2	Controllable by skilled driver	situation is conrollable by driver with training	QM
B5	Lane change on highway or freeway, strong traffic	Airbag inflates, false positive	Driver is shocked, looses control over car	S3	Life-threatening injuries (survival uncertain), fatal injuries	life threatening or fatal injuries	E4	High probability	Lane change on highway is very frequent	C3	Difficult to controll or uncontrollable by skilled driver	situation is difficult to control for a skilled driver	D

Bild 2: Auszug der Analyse zur Bestimmung der ASIL. (© MethodPark)

„False Positive“ und eines „False Negative“ zu beleuchten – also das Fehlverhalten des Systems dahingehend, dass eine Aktion ungewollt beziehungsweise nicht ausgeführt wird. Für FABSY heißt das konkret:

1. Was kann passieren, wenn der Airbag ungewollt auslöst?
2. Was kann passieren, wenn der Airbag nicht auslöst, obwohl er sollte?

Da sich die Auswirkungen dieser beiden Fehlerfälle je nach Fahrsituation und Umgebungsbedingungen unterscheiden, gilt es diese ebenso zu berücksichtigen wie die Schwere eines Unfalls und die Fähigkeit des Fahrers diese Situation zu beherrschen.

Für das fiktive Beispielsystem ist das kritischste Szenario zu Fragestellung 1. demnach: Bei einem Spurwechsel auf der Autobahn in dichtem Verkehr löst der Fußgänger-Airbag fälschlicherweise aus. In dem betrachteten Szenario würde ein überraschter und erschrockener Fahrer die Kontrolle über sein Fahrzeug verlieren und einen Unfall verursachen.

Gemessen an der Gesamtzahl der Verkehrssituationen „Spurwechsel des Fahrzeugs auf der Autobahn bei dichtem Verkehr“ ist das Fahrzeug wahrscheinlich häufiger in dieser Situation, selbst ein geübter Fahrer kann die Situation mit hoher Wahrscheinlichkeit nicht kontrollieren, und die Folgen für den Fahrer, Insassen und andere Verkehrsteilnehmer sind mit hoher Wahrscheinlichkeit fatal.

Aus diesen eben genannten Faktoren von Unfallschwere (engl. Severity = S), Kontrollierbarkeit (engl. Controllability = C) und Auftretenswahrscheinlichkeit (engl. Probability of Exposure = E) lässt sich mithilfe des in Bild 1 dargestellten Risikographen die Kritikalitätsstufe ASIL D (Automotive Safety Integrity Level) bestimmen.

Ein mögliches Szenario für Fragestellung 2: Bei schnellem Fahren durch die Stadt läuft ein Fußgänger auf die Fahrbahn, es kommt zum Zusammenstoß, aber der Front-Airbag löst nicht aus. Für den Fußgänger kann dies lebensbedrohlich sein, und für den Fahrer ist die Situation so gut wie nicht kontrollierbar. Analog dem Vorgehen von Fragestellung 1 erhält man in diesem Szenario mithilfe des Risikographen die Kritikalitätsstufe ASIL B.

Bild 2 zeigt einen Auszug aus der Analyse zur Bestimmung des ASIL und fasst die »

i Herleitung der Echtzeitanforderungen

Zwar ist die Detektion eines Hindernisses bereits aus einiger Entfernung möglich, jedoch oft nicht sinnvoll, da sich das Hindernis noch aus dem Fahrweg entfernen kann, wie etwa ein Passant, der in einiger Entfernung die Straße quert. Selbst wenn das Hindernis im Fahrweg bleibt, kann eine autonome Notbremsung genügen, um eine Kollision zu vermeiden. Kraftfahrzeuge müssen heute eine Bremsverzögerung von mindestens 5,0 m/s² erreichen, oft sind Werte von bis zu 10 m/s² möglich, sodass im städtischen Verkehr (50 km/h, also 14 m/s) eine Vollbremsung das Fahrzeug innerhalb von 2 Sekunden zum Stillstand bringen kann. Zusammen mit der Annahme, dass ein Fußgänger zum Queren eines Fahrstreifens ca. 5 Sekunden benötigt (aufgeteilt in Betreten und Verlassen der Fahrbahn mit je 2,5 Sekunden), ergibt sich eine Oberschranke von ca. 2 Sekunden, ab der das System keinen Nutzen mehr bringt.

Mit zunehmender Geschwindigkeit verschiebt sich die Lage jedoch dramatisch: Bei 100 km/h kann eine autonome Notbremsung das Fahrzeug innerhalb von 2 Sekunden auf bestenfalls 40 km/h abbremsen, in der Realität werden jedoch selbst auf trockener Fahrbahn erheblich geringere Verzögerungen erzielt.

Daher muss das System die Auslösungsnotwendigkeit trotz Notbremsung mehrfach innerhalb der zeitlichen Oberschranke von 2 Sekunden analysieren. Da eine spätestmögliche Situationsanalyse wünschenswert und in der Wirkkette nachgelagert noch eine volle Expansion des Airbags nötig ist, die bis zu 50 Millisekunden dauern kann, ergibt sich eine realistische Echtzeitanforderung von 150 Millisekunden vom Auftreten des Hindernisses bis zur Signalisierung des Airbag-Zünders.

eben beschriebenen Szenarien in tabellarischer Form nochmals zusammen.

Wie oben bereits gezeigt, müssen nun aus den Ergebnissen der Gefahren- und Risikoanalyse die Sicherheitsziele formuliert werden. In unserem Fallbeispiel könnten diese wie folgt aussehen:
Sicherheitsziel 1: Verhindere ungewolltes Auslösen des Airbags (ASIL D).
Sicherheitsziel 2: Wenn eine Kollision mit einem Fußgänger unvermeidbar ist, garantiere das Auslösen des Airbags (ASIL B).

Funktionale Architektur

Ein weiterer Schritt hin zur Erstellung des funktionalen Sicherheitskonzepts ist die Erarbeitung der funktionalen Architektur. Im Kontext der ISO 26262 wird die funktionale Architektur mit dem Begriff „preliminary architecture" bezeichnet, wir werden im Folgenden jedoch den auch in anderen Domänen etablierten Begriff „funktionale Architektur" verwenden. In der funktionalen Architektur erfolgt eine Dekomposition von FABSY aus rein funktionaler Sicht: Sie beschreibt eine Zerlegung des Systems in Funktionsblöcke und deren Zusammenwirken. FABSY benötigt als Funktionsblöcke beispielsweise die Akquisition eines Bildes, das Erkennen eines Fußgängers auf diesem Bild, die Vorhersage einer Kollision und schließlich die Entscheidung den Airbag auszulösen.

Dabei abstrahiert die funktionale Systemarchitektur von der konkreten, technischen Realisierung. Es wird zunächst offengelassen, auf welchem Steuergerät und mit welcher Technologie ein bestimmter Funktionsblock realisiert werden soll. So bleiben verschiedene Varianten für die technische Systemarchitektur möglich und können später systematisch verglichen werden. Bild 3 zeigt die initiale, funktionale Architektur für FABSY.

Aus der Gefahren- und Risikoanalyse leiten sich Safety-Anforderungen ab, die den Funktionsblöcken zugeordnet werden. Dabei vererbt sich die ASIL-Einstufung des Sicherheitsziels auf die Safety-Anforderungen und weiter auf die Funktionsblöcke. So ergibt sich zum Beispiel für den Funktionsblock „Detect Person" ein ASIL D, da die fälschliche Erkennung einer Person das Sicherheitsziel 1 verletzen würde.

Eine Funktion, wie die Personenerkennung, lässt sich über eine einzige Technologie, wie etwa Kamera oder Radar, nicht mit der für ASIL D erforderlichen Zuverlässigkeit gewährleisten. Bereits in der funktionalen Architektur kann darauf mit der Architekturmaßnahme „ASIL Dekomposition" reagiert werden: Die Kette von der Akquisition des Bildes bis zur Entscheidung den Airbag auszulösen wird redundant mit zwei unterschiedlichen Technologien realisiert. Diese Art der Redundanz wird auch als funktionale Redundanz bezeichnet. Die Auslösung des Airbags erfolgt nur, wenn radarbasierte und kamerabasierte Erkennung zur gleichen Entscheidung kommen. Unter der Voraussetzung, dass sich die radar- und kamerabasierten Teilsysteme nicht ungewollt gegenseitig beeinflussen können (d.h. Rückwirkungsfreiheit, engl. „freedom from interference"), lässt sich so der ASIL dieser Funktionsblöcke von ASIL D auf ASIL B (D) reduzieren. Funktionsblöcke, für die keine Dekomposition vorgenommen wird, behalten ihre ursprüngliche ASIL-Einstufung.

Bild 4 zeigt die funktionale Systemarchitektur nach der ASIL Dekomposition.

Bereits in der funktionalen Systemarchitektur werden die für die Sicherheitsziele relevanten Wirkketten modelliert und den Wirkketten End-to-End-Echtzeitanforderungen zugewiesen.

So ergibt sich aus der Analyse der Systemfunktion beispielsweise, dass zwischen dem Eintritt eines Fußgängers in den Fahrweg und dem Auslösen des Airbags maximal 150 ms liegen dürfen.

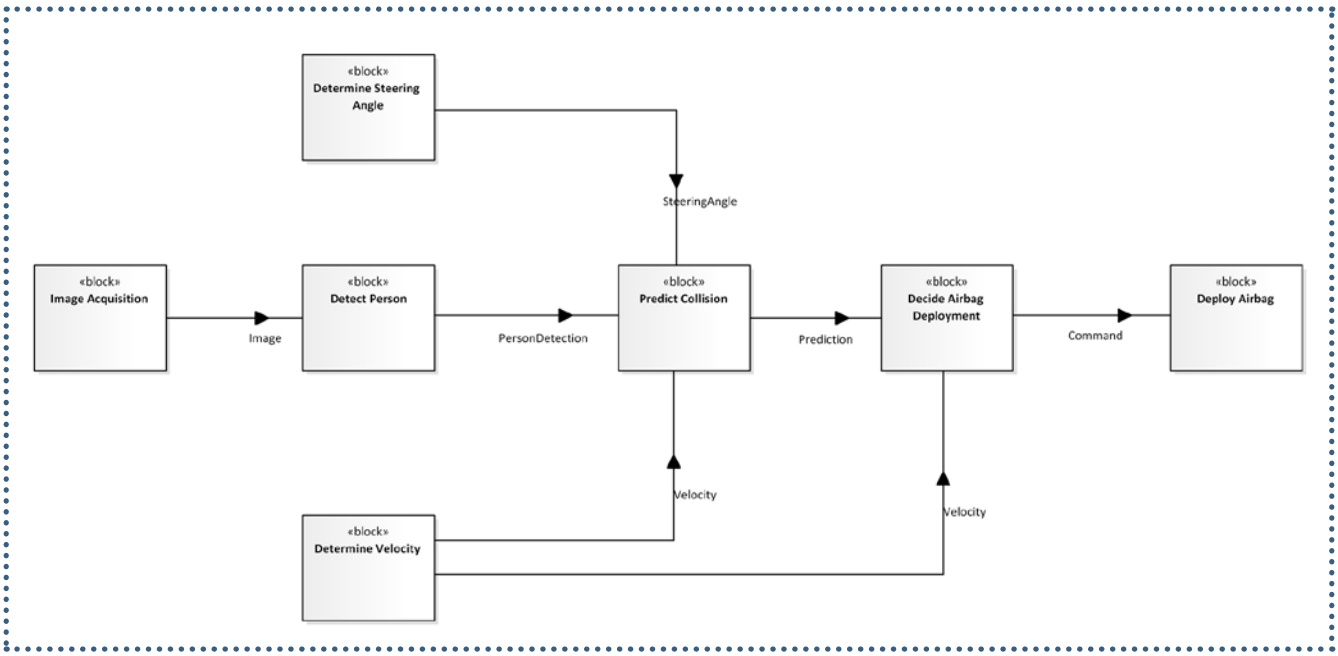

Bild 3: Funktionale Systemarchitektur von FABSY. (© MethodPark)

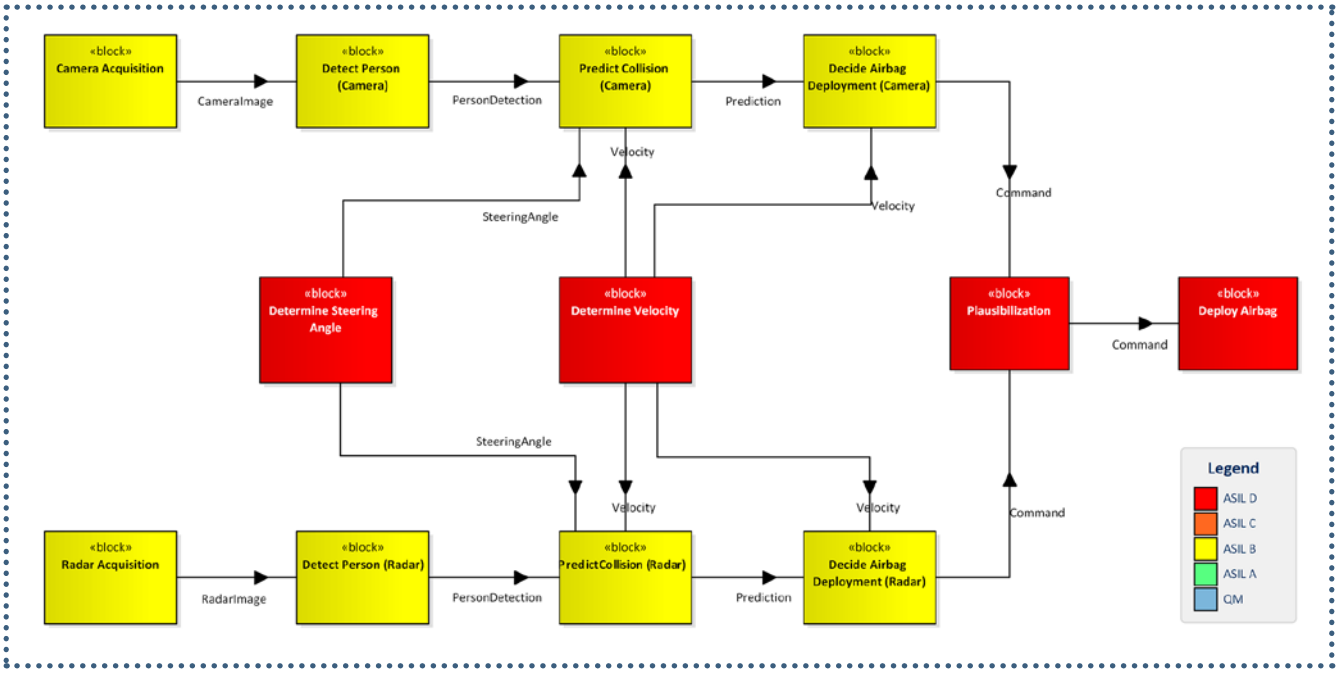

Bild 4: Funktionale Systemarchitektur von FABSY nach der ASIL Dekomposition. Den Funktionsblöcken ist ein ASIL B (gelb) bzw. ASIL D (rot) zugewiesen. (© MethodPark)

Technische Architektur

Die technische Architektur ist Grundlage für das technische Sicherheitskonzept. In der technischen Architektur wird die konkrete technische Realisierung der funktionalen Architektur beschrieben. Dies umfasst unter anderem die Entscheidung, welche Steuergeräte beteiligt sind, wie diese miteinander vernetzt und wie die Funktionsblöcke auf die verschiedenen Steuergeräte verteilt werden.

Dieser Ansatz integriert sich nahtlos mit der Vorgehensweise im AUTOSAR-Standard, der eine flexible Zuordnung von Software-Komponenten zu Steuergeräten ermöglicht.

Im vorherigen Abschnitt zur funktionalen Architektur wurde bereits die Rückwirkungsfreiheit angesprochen, die nun in der technischen Architektur hergestellt werden muss. Sie ist Grundlage für Redundanztechniken und die ASIL Dekomposition. Sie bezieht sich auf drei Aspekte, die auch in Teil 6 der ISO 26262 aufgelistet werden:

- Räumliche Isolation der Komponenten,
- Zeitliche Isolation der Komponenten,
- Sicherer Austausch von Informationen zwischen den Komponenten.

Isolation bedeutet, dass sich mögliche Fehler einer Komponente nicht auf andere Komponenten ausbreiten können. Ein zeitlicher Fehler ist zum Beispiel die fälschliche Monopolisierung des Prozessors durch einen Task. Ein anderer sicherheitskritischer Task, der ebenfalls an diesen Prozessor gebunden ist, würde in seiner Ausführung behindert werden, was möglicherweise ein Sicherheitsziel gefährdet. Die Isolationsdomänen können sowohl durch physische Trennung als auch durch software- und hardwarebasierte Techniken hergestellt werden. Die Auswahl geeigneter Mechanismen liegt im Bereich der Architekturentscheidungen in den verschiedenen Disziplinen System, Software und Hardware. Beispielsweise kann ein Watchdog-Baustein auf einem Steuergerät verwendet werden, um zeitliche Isolation zu gewährleisten; er dient damit als Sicherheitsnetz. Andererseits kann eine Funktion auch auf einem dedizierten Steuergerät ausgeführt werden und somit die gesamte Rechenzeit für sich beanspruchen. Die Entscheidung für die eine oder andere Variante wird durch die funktionalen und nichtfunktionalen Eigenschaften der jeweiligen Lösungen sowie durch die Betrachtungen der funktionalen Sicherheit im Rahmen der Failure-Mode-and-Effects Analyse (FMEA) getroffen. Im Folgenden nehmen wir an, dass diese Analysen bereits durchgeführt wurden.

Im Rahmen des Fallbeispiels werden nun exemplarisch eine integrierte Architektur und eine föderierte Architektur als Alternativen betrachtet:

- Variante „Dedizierte ECUs“ (Bild 5): In dieser föderierten Architektur werden die kamera- und radarbasierten Pfade auf den dedizierten Steuergeräten Camera ECU und Radar ECU ausgeführt. Die Plausibilisierung der beiden Auslöse-Entscheidungen und die eigentliche Auslösung des Airbags erfolgen auf der FABSY ECU. Verwendet man dedizierte ECUs, dann können die beiden Pfade hardwareseitig sehr gut isoliert werden.
- Variante „CDAC ECU“ (Bild 6): In dieser integrierten Architektur wird ein leistungsfähiges Steuergerät CDAC (Central Driver Assistance Controller) eingesetzt. Die gesamte Verarbeitung von der Akquisition des Bildes bis zur Entscheidung über die Auslösung des Airbags wird auf der CDAC ECU ausgeführt, und zwar für den kamera- und den radarbasierten Pfad. Die Plausibilisierung der Entscheidungen und die Auslösung des Airbags erfolgen wie in der Variante „Dedizierte ECUs“ auf der FABSY ECU. Die Konsolidierung der Radar- und Kameraverarbeitung sowie evtl. weiterer Fahrer-Assistenzfunktionen auf »

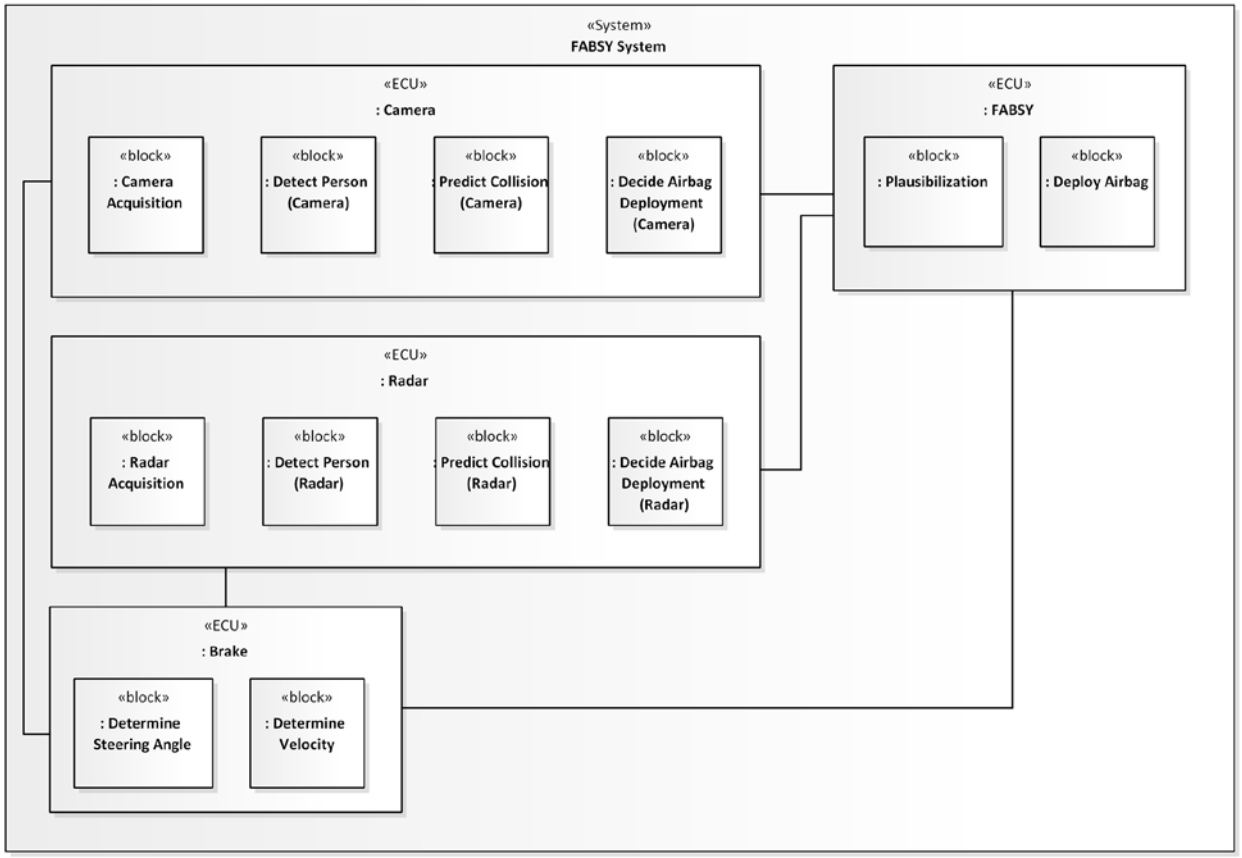

Bild 5: Allokation der Funktionsblöcke in der Technischen Systemarchitektur, Variante „Dedizierte ECUs“. (© MethodPark)

einem leistungsfähigen Steuergerät kann Vorteile bezüglich Kosten, Gewicht und Bauraum haben, erfordert jedoch Maßnahmen in der System- und Software-Architektur zur Isolation der Pfade.

Die technischen Architekturvarianten können nun bezüglich der Erfüllung der Echtzeitanforderungen und weiterer Kriterien, wie etwa der Hardware-Kosten, bewertet werden. Für die Umsetzbarkeit einer preislich günstigeren Variante ist jedoch entscheidend, dass alle Echtzeitanforderungen erfüllt werden können. Weiterhin trägt die Einhaltung der Echtzeitanforderungen auch zur Herstellung der zeitlichen Isolation bei.

Echtzeitanforderungen

Im Rahmen des FABSY Beispiels müssen harte Echtzeitanforderungen erfüllt werden. Dies bedeutet, dass es einen sogenannten Termin (engl. deadline) gibt, also einen spezifizierten Zeitpunkt, an dem beispielsweise ein Task seine Aufgabe garantiert abgearbeitet haben muss. Wird der Termin nicht eingehalten, hat das massive Auswirkungen auf die Systemfunktion und auf die mit ihr verbundenen Sicherheitsziele. Auch die Latenzzeit spielt eine Rolle, d. h. die Zeitdauer zwischen Auftreten eines Ereignisses und eines späteren, erwarteten Folgeereignisses. Letzteres kann beispielsweise der Beginn der Abarbeitung von Radar- oder Kameradaten oder das Senden eines Signals zum Feuern des Airbags sein. Folgende Echtzeitanforderungen müssen eingehalten werden und dienen als Basis für die Bewertung der beiden Varianten:

- Das korrekte Scheduling muss sichergestellt und Mehrfachaktivierungen verhindert werden.
- Die Latenzzeit zwischen dem Eintritt eines Fußgängers in den Fahrweg des Autos und dem Auslösen des Airbags darf maximal 150 ms betragen.
- Die Latenzzeit zwischen dem Empfang eines Kamerabildes und des zur Plausibilisierung verwendeten Radarbildes darf maximal 75 ms betragen.
- Bei jedem Auftreten eines Objektes in einer definierten Entfernung muss der Airbag ausgelöst werden (keine Abrisse in der Verarbeitungskette).

Durch eine modellbasierte Analyse des dynamischen Verhaltens lässt sich frühzeitig sicherstellen, dass keine systematischen, zeitlichen Fehler in einer Architektur enthalten sind. Hierfür wird ein Timing-Modell erstellt, das die Eigenschaften der jeweils untersuchten Architekturebene berücksichtigt, die für das dynamische Verhalten relevant sind. Bei einem ASIL-D-System ist zwar der Einsatz eines externen, zeitlichen Überwachungsmechanismus nötig (etwa externer Watchdog), der als Sicherheitsnetz fungiert, da andere systematische und zufällige Fehler Laufzeitanomalien verursachen können. Allerdings werden durch die Einhaltung der Echtzeitanforderungen mittels des Timing-Modells systematisch Fehler vermieden, die ihren Ursprung in der unzureichenden Betrachtung des dynamischen Verhaltens haben, beispielsweise unzureichende Bereitstellung von Rechenzeit für die Software oder eine Überschreitung der maximalen Zeitdauer für die Inter-ECU oder Inter-Core Kommunikation. Durch diese Fehler-Vermeidungstechnik steigt die Verfügbarkeit des

Systems; sie kann einen Beitrag zur Realisierung von fail-operational Systemen leisten.

Analyse auf unterschiedlichen Architekturebenen

Eine erste Bewertung der Varianten kann bereits auf der Ebene der funktionalen Architektur erfolgen, d. h. unabhängig von der konkreten Hardware und unabhängig vom Scheduling der Tasks und Unterbrechungsbehandlungen (engl. Interrupt Service Routinen, ISR) auf einer CPU. Hierzu werden für die einzelnen Funktionsblöcke Zeitbudgets und Aktivierungen (periodisch, sporadisch) vorgegeben. Da auf dieser Architekturebene keine Scheduling-Effekte berücksichtigt werden, sind die Zeitbudgets als Bruttozeiten vorzugeben, d. h. einschließlich aller Verdrängungen durch andere Tasks und ISRs.

Bereits mit einem solchen hardware-unabhängigen Timing-Modell ist es möglich, die Anforderungen 2 bis 4 zu überprüfen. Beginnt man mit der Analyse auf der Ebene der funktionalen Architektur, besteht der Vorteil darin, den Kamera- und Radarpfad als Wirkkette(n) sowie die Messpunkte für die Anforderungen 2 bis 4 eindeutig zu beschreiben. Dies ist erfahrungsgemäß ein sehr wichtiges, aber nicht einfaches Unterfangen, da hierfür Fachleute aus unterschiedlichen Bereichen beteiligt sein müssen.

Eine sich anschließende detaillierte Analyse der technischen Architektur erfolgt hardware-abhängig, indem u. a. folgende Eigenschaften in einem Timing-Modell berücksichtig werden:

- Busse (hier FlexRay und CAN)
- Steuergeräte (ECU) und Prozessoren (hier Single- und Multi-Core)
- Zuordnung von Software-Komponenten auf ECUs, CPUs und Cores
- Tasks, Mapping von Tasks/ISRs auf CPUs und Cores
- Scheduling-Eigenschaften von Tasks und ISRs (Scheduling-Verfahren, Prioritäten, Unterbrechbarkeit etc.)

Die ersten drei Punkte müssen gemäß ISO 26262 Part 4 bei einer Verifikation des Systementwurfs berücksichtigt werden, Punkt vier und fünf gemäß ISO 26262 Part 6 bei der Verifikation des Software-Entwurfs.

Eine Simulation als geeignete Methode zur Verifikation des Systementwurfs ist gemäß des Standards für ASIL A und B „recommended" und für ASIL C und D „highly recommended". Für eine Verifikation des Software-Entwurfs wird im Standard neben der Simulation (ASIL A bis C: „recommended", ASIL D: „highly recommended") auch auf die Analyse des Datenflusses hingewiesen (ASIL A und B: „recommended", ASIL C und D: „highly recommended"). In der Datenflussanalyse des FABSY Beispiels wurden die bereits oben beschriebenen Anforderungen (Anforderungen 2 bis 4) an die Wirkketten simulativ untersucht.

Im Folgenden wird auf die Analyse, Optimierung und Verifikation während des Software-Entwurfs für die beiden Varianten der technischen Architektur eingegangen.

Bewertung der beiden Varianten

Häufig treten beim ersten Entwurf einer System- oder Software-Architektur Echtzeitfehler auf, sodass eine Optimierung »

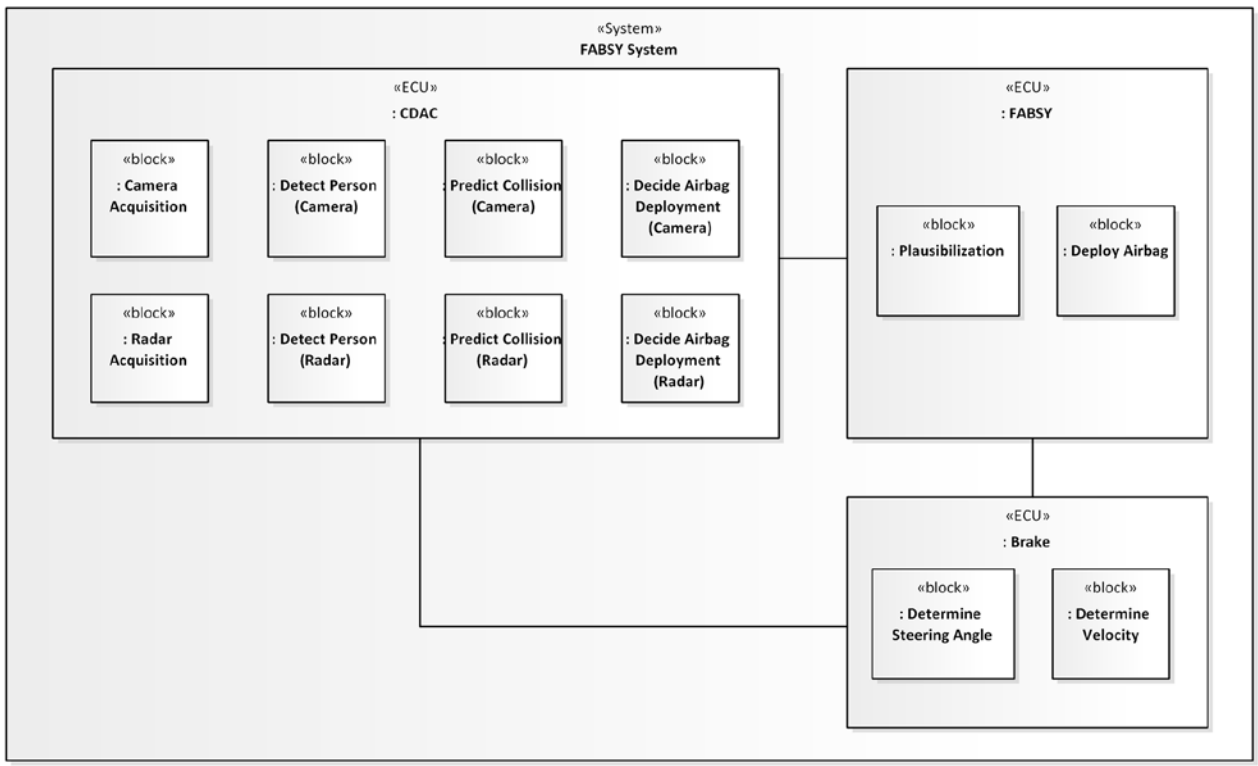

Bild 6 Allokation der Funktionsblöcke in der Technischen Systemarchitektur, Variante „CDAC ECU". (© MethodPark)

notwendig ist. Die Zuordnung von Software-Komponenten und Tasks/ISRs sowie die Scheduling-Eigenschaften sind auf der Ebene der Software-Architektur Freiheitsgrade, die typischerweise in einem Entwicklungsprozess relativ einfach geändert werden können. Gleichzeitig haben diese Eigenschaften der Software-Architektur einen wesentlichen Einfluss auf die Einhaltung der Echtzeitanforderungen.

Eine Festlegung dieser Freiheitsgrade ist Bestandteil des Entwurfsprozesses und sollte in einem effizienten Entwicklungsprozess modellbasiert analysiert und optimiert werden. D.h. die Erstellung eines Timing-Modells, die Verifikation der Anforderungen und ggf. notwendige Optimierungen der Architektur sind ein integraler Bestandteil des Entwicklungsprozesses; sie werden nicht nur für Sicherheitsanforderungen durchgeführt.

Weitere Freiheitsgrade sind beispielsweise, von welchen Cores aus auf die Peripherie und Busse zugegriffen werden können oder die Implementierung der einzelnen Module der AUTOSAR Basic Software.

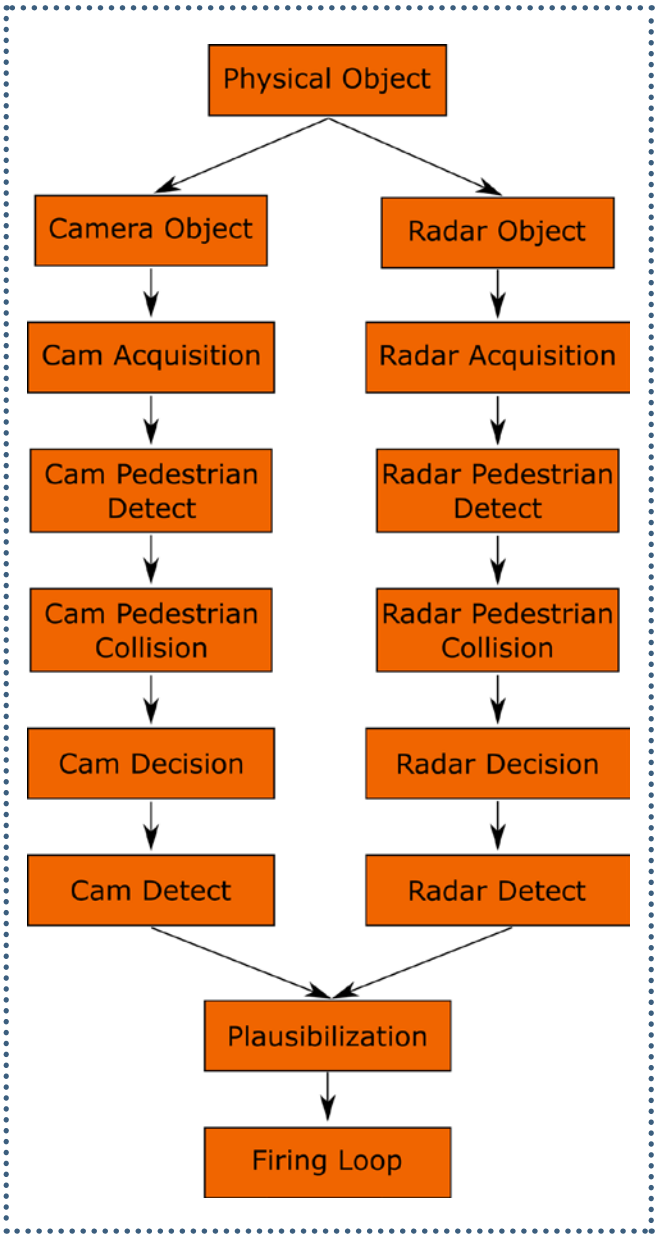

Bild 7: Spezifikation der Kamera- und Radarwirkkette.
(© MethodPark)

Um bei der detaillierten Analyse das Scheduling der Tasks und ISRs zu berücksichtigen, werden die Ausführungszeiten der Tasks als Nettozeiten spezifiziert, d.h. es handelt sich um die „reine" Ausführungszeit der Task ohne Unterbrechung durch andere Tasks oder ISRs. Häufig können die Netto-Ausführungszeiten der Software während der Entwurfsphase nicht auf dem Zielprozessor gemessen werden, da eine Implementierung noch nicht vorliegt. In diesem Fall werden die Netto-Ausführungszeiten als Zeitbudgets festgelegt. Grundlage hierfür können Vorgaben, Expertenschätzungen oder Messungen aus Vorprojekten sein.

Ein mit den oben beschriebenen Eigenschaften der technischen Architektur erstelltes Timing-Modell ist die Basis, um mit einem geeigneten Analysewerkzeug die Anforderungen noch vor einer Implementierung zu verifizieren (Verification of system design, ISO 26262-4). Hierbei ist die Verifikation von Anforderung 1 (korrektes Scheduling, keine Mehrfachaktivierungen) ein direktes Ergebnis einer Scheduling-Analyse. Hingegen wurden für die weiteren Anforderungen (2 bis 4) Wirkketten modelliert, die abstrakt den Datenfluss beschreiben. Hierbei ist jeder Wirkkettenschritt ein relevanter Kommunikationspunkt, an dem Daten gepuffert, gelesen und anschließend verarbeitet oder geschrieben werden. Für jedes auslösende Ereignis (bei Anforderung 2 ist dies das Auftreten eines Objektes) wird während der Simulation eine neue Instanz der Wirkketten erzeugt und seine Anforderungen verifiziert.

Durch dieses Vorgehen ist es möglich, das dynamische Verhalten schon vor der Implementierung zu simulieren. Der dynamische Anteil der Architektur wird so vergrößert und die Abdeckung bei „Simulation of dynamic parts of the design" erhöht. Dieser Ansatz lässt sich sowohl auf die einfachen Modelle anwenden, die nur aus den Blöcken der technischen Architektur bestehen, als auch auf Modelle, die bereits Tasks und ISRs enthalten.

Bild 7 zeigt exemplarisch die Spezifikation der Wirkkette für Anforderung 2 vom Auftreten des Objektes bis zum Senden des Kommandos an den Airbag. In Bild 8 ist die Analyse des dynamischen Verhaltens als Ausgabe des Echtzeitsimulators chronSIM in einem Gantt Chart für die Variante „Dedizierte ECUs" zu sehen. Dargestellt ist neben den RTOS-Zuständen der Tasks und ISRs (ausgefüllte Farbe symbolisiert den Zustand „running") das Senden von Signalen auf dem CAN- und FlexRay-Bus sowie eine Instanz der Wirkkette (ockerfarbene Linien) von Anforderung 2 in Abhängigkeit der Zeit.

Im ersten Schritt der Wirkkette erscheint ein physikalisches Objekt und wird verzögert von der Kamera und dem Radar erfasst. Anschließend werden die Radar- (oberer Teil der Wirkkette) und die Kameradaten (unterer Teil der Wirkkette) jeweils verarbeitet, über einen CAN- bzw. FlexRay-Bus zur FABSY ECU gesendet und dort plausibilisiert. In diesem konkreten Fall wird die maximale Latenzzeit der Wirkkette in Höhe von 150 ms eingehalten.

Das bisher vorgestellte methodische Vorgehen erlaubt eine effiziente Optimierung der technischen Architektur. Dieses Vorgehen führt bereits nach wenigen Optimierungs-

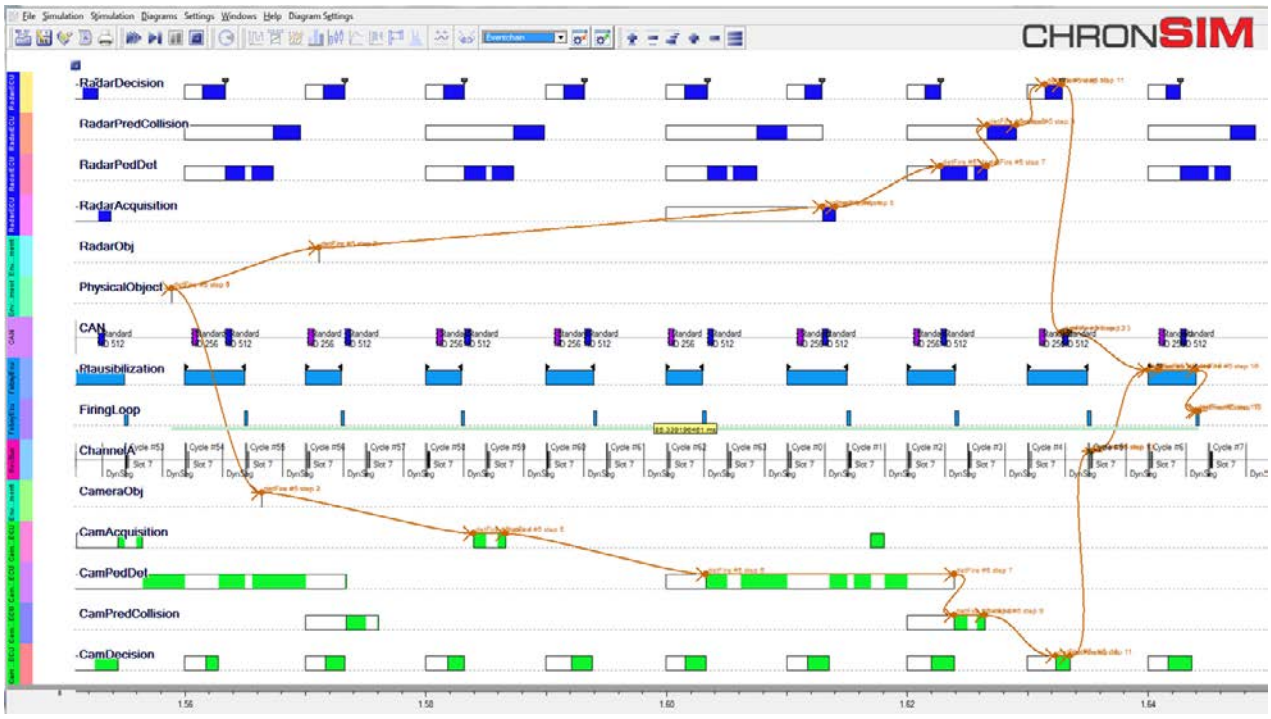

Bild 8: Analyse der Radar- und Kamerawirkkette für die Variante „Dedizierte ECUs“ einschließlich der Plausibilisierung mit dem Echtzeitsimulator chronSIM. (© MethodPark)

schleifen dazu, dass beide Varianten („Dedizierte ECU“ und „CDAC ECU“) ihre Echtzeitanforderungen einhalten. So kann man anhand weiterer (auch nichttechnischer) Kriterien, wie etwa Kosten oder Hardware-Lieferantenbeziehung, eine Auswahl treffen.

Im Folgenden werden weitere Analysen aufgezeigt, die in einer späteren Projektphase nach der Implementierung der Software durchgeführt werden können.

Weiteres Vorgehen im Entwicklungsprozess

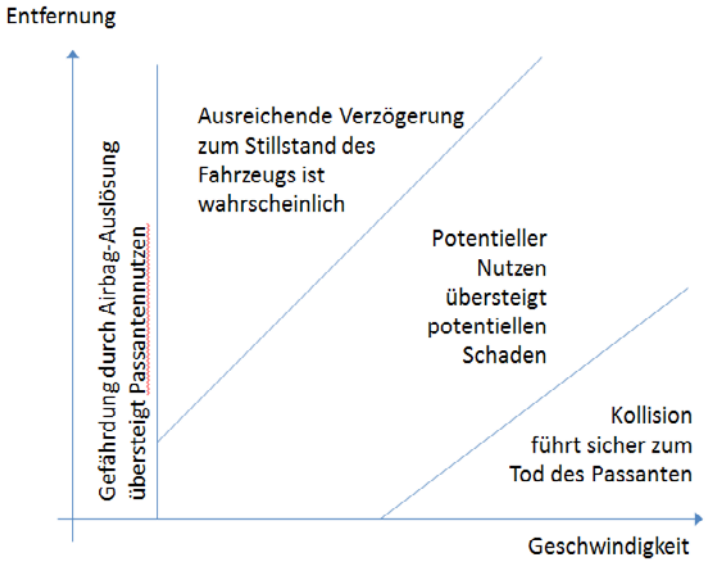

Bild 9: Herleitung der Echtzeitanforderung. (© MethodPark)

Die hier beschriebene Analyse ist Teil der quantitativen Analyse zur Bewertung von Echtzeiteigenschaften in einer oder mehreren Architekturen. Die Analyse-Ergebnisse bilden eine solide Ausgangsbasis für die weitere Umsetzung und für Architekturdurchstiche. Im Rahmen der iterativ-inkrementellen Entwicklung fließen Informationen und Rückmeldung aus der Implementierung in den Architekturentwurf ein.

Während der iterativ-inkrementellen Entwicklung können die vorgegebenen Zeitbudgets durch Netto-Ausführungszeiten und die Aktivierung der Tasks und ISRs (sowie Runnables und Mainfunctions) auf der Basis von Messdaten konkretisiert werden.

Die Netto-Ausführungszeiten können durch eine statische, semantische Code-Analyse auf Implementierungsebene ermittelt werden. Hierfür wird ein Modell der Hardware benötigt, das beispielsweise die Abarbeitungsdauer von Prozessor-Instruktionen, die Anzahl und Anbindung der Prozessoren, Pipelining-Effekte oder die Speicherarchitektur berücksichtigt. Dieses Hardware-Modell kann eingesetzt werden, um die Netto-Ausführungszeit des Source-Codes zu ermitteln. Der Einsatz dynamischer, messbasierter Verfahren auf dem ausgeführten Programm rundet das Bild weiter ab. Allerdings muss hierbei genau analysiert werden, ob in der Messung auch die längsten und kürzesten Ausführungspfade des Programms zum Tragen kommen.

Bei Multi-Core Prozessoren ist zu berücksichtigen, dass durch die parallele Ausführung von Code gegenüber Single-Core Prozessoren neue Arten von Konkurrenzsituationen von gemeinsam genutzten Ressourcen entstehen, wie z. B. Speicher oder Peripherie. Die Annahme, dass immer Ressourcenkonflikte auftreten, führt zwar zu sicheren Aussa- »

gen über das Echtzeitsystem, jedoch sind sie in Bezug auf die Umsetzung der Funktionalität oftmals zu pessimistisch.
Der vorgestellte Top-Down Spezifikationsansatz, der die FABSY-spezifischen Task-Eigenschaften und das Scheduling betrachtet, kann problemlos durch eine Bottom-Up Analyse ergänzt werden: So kann man einerseits mit verschiedenen Konfigurationen zur Ressourcenverteilung experimentieren, wie beispielsweise das Memory Mapping oder die Allokation von Tasks zu Cores. Bezieht man andererseits anwendungsspezifische Informationen ein, lassen sich optimistischere Aussagen über die Echtzeiteigenschaften des Programms in Ausführung auf einer konkreten Hardware treffen.

Diese ganzheitliche Analyse erfolgt iterativ, um das Echtzeitmodell schrittweise zu verfeinern. Dadurch kann man auf kostengünstige Art und Weise Software- und Hardware-Konfigurationen und Implementierungen unterschiedlich miteinander kombinieren. Dieser Ansatz beeinflusst Sicherheitsanforderungen, Qualitätseigenschaften sowie Kostenbetrachtungen positiv.

Fazit

Das vorgestellte Vorgehen ermöglicht niedrigere Entwicklungskosten durch frühzeitige und nachhaltige Architekturentscheidungen im Kontext von ISO 26262 und harter Echtzeit. Der Kernpunkt ist eine ganzheitliche Betrachtung von Echtzeit- und Safety-Anforderungen einschließlich von Wirkketten sowie eine Verifikation der Anforderungen mit einem geeigneten Analysewerkzeug, wie z. B. dem Echtzeitsimulator chronSIM. Eine solche Verifikation sollte bereits auf der Ebene der hardware-unabhängigen funktionalen Architektur beginnen. Hardware-Abhängigkeiten der technischen Architektur werden während der Entwicklung iterativ bei der Verifikation der Anforderungen berücksichtigt. Der beschriebene Ansatz beeinflusst Sicherheitsanforderungen, Qualitätseigenschaften sowie Kostenbetrachtungen positiv. Er lässt sich effizient in einen Entwicklungsprozess integrieren.

» **www.methodpark.de**

» **www.hanser-automotive.de/4334179**

Hier finden Sie die Download-Version des Beitrags.

Dr. Ulrich Becker, **Christian Lederer** und **Frank Pinecker** sind bei Method Park als Consultants, Trainer und Coach tätig. **Dr. Isabella Stilkerich** ist als Software-Architektin im ARAMiS 2 Forschungsprojekt (BMBF Förderkennzeichen 01IS16025) bei Schaeffler Technologies tätig. **Dr. Ralf Münzenberger** ist Mitgründer der INCHRON GmbH. **Philip Rehkop** arbeitet bei INCHRON als Professional Services Engineer.